VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 Emília Chovancová

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ANALÝZA MOŽNÝCH HROZEB NA STAROU VERZI OS LINUX

ANALYSIS OF POSSIBLE THREATS IN OLD OS LINUX VERSION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Emília Chovancová**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Adrián Tomašov**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Studentka: Emília Chovancová ***ID:*** 203705

Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Analýza možných hrozeb na starou verzi OS Linux

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je analyzovat možné útoky na OS Linux se starší verzí jádra a běžících služeb a následně navrhnout způsoby ochrany vůči možným útokům. Systém bude připojený do veřejné sítě a lze předpokládat, že bude podléhat různým útokům. V úvodníi části práce se zaměří na studium různých typů útoků, které lze aplikovat na sledovaný systém. Praktickým výstupem práce je aplikace útoků na zařízení se zastaralým softwarem. Následně se student zaměří na návrh a implementaci ochranných prostředků vůči útokům (například přidáním firewallu, úprava konfigurací služeb systému nebo nasazení aktivních ochranných prostředků), a otestuje jejich účinnost.

DOPORUČENÁ LITERATURA:

[1] CHEN, Haogang, Yandong MAO, Xi WANG, Dong ZHOU, Nickolai ZELDOVICH a M. Frans KAASHOEK. Linux kernel vulnerabilities. In: Proceedings of the Second Asia-Pacific Workshop on Systems - APSys '11. New York, New York, USA: ACM Press, 2011, 2011, s. 1-. ISBN 9781450311793. Dostupné z: doi:10.1145/2103799.2103805

[2] STUCKMAN, Jeffrey a James PURTILO. Mining Security Vulnerabilities from Linux Distribution Metadata. In: 2014 IEEE International Symposium on Software Reliability Engineering Workshops [online]. IEEE, 2014, 2014, s. 323-328 [cit. 2020-09-15]. ISBN 978-1-4799-7377-4. Dostupné z: doi:10.1109/ISSREW.2014.101

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 31.5.2021

Vedoucí práce: Ing. Adrián Tomašov

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Bakalárska práca sa sústreďuje na bezpečnostný aspekt staršej verzie operačného systému Linux. Dokument je rozdelený na teoretickú časť, ktorá obsahuje opis operačného systému, jadra, GNU/LINUX a Linux operačného systému. Ďalej sa opisujú aktívne a pasívne útoky a ich príklady, teda replay, maškaráda, denial of service, man in the middle a odpočúvanie a analyzovanie premávky v sieti. Nasledujúca časť pojednáva o potenciálnych zraniteľnostiach, ktoré sú na operačných systémoch bežné. Pomocou skenovacích softwérov Nmap a Lynis sa pokračuje do praktickje časti, ktorá obsahuje rôzne druhy úspešných a aj neúspešných útokov ako password recovery, denial of service, spectre, brute force použitá proti heslu a taktiež cross-site scripting. Posledná časť dokumentu obsahuje ansible skript, ktorý pokrýva zabezpečenie systému.

KĽÚČOVÉ SLOVÁ

Linux, OS, zabezpečenie, zastaralý systém, útoky, DoS, MITM, cross-site skript, brute force, ansible, fail2ban, skript

ABSTRACT

The bachelor thesis is focused on the security aspect of an older version of a Linux based machine. The document is split into a theoretical part which contains a description of what operating system, kernel, UNIX, GNU/LINUX and Linux OS are in general. Then the description proceeds to explanation of active and passive attacks such as replay, masquarade, denial of service, man in the middle or listening and analysing the network. Afterwards it continues with description of potentially vulnerable spots that are common on operating systems. With the help provided from scanning software, especially Nmap and Lynis, the document proceeds to the practical part which contains various types of successful and unsuccesfful attacks such as password recovery, denial of service, spectre, brute force of a password and cross-site scripting. The last part covers the protection against succesfull attacks and adds a little bit more of additional protection in general in a form of an ansible script.

KEYWORDS

Linux, OS, securing, old system, attacks, DoS, MITM, cross-site scripting, brute force, ansible, fail2ban, script

CHOVANCOVÁ, Emília. *Analýza možných hrozeb na starou verzi OS Linux*. Brno, 2020, 46 s. Bakalárska práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Ing. Adrián Tomašov

VYHLÁSENIE

Vyhlasujem, že svoju bakalársku prácu na tému „Analýza možných hrozeb na starou verzi OS Linux“ som vypracovala samostatne pod vedením vedúceho bakalárskej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autorka uvedenej bakalárskej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušila autorské práva tretích osôb, najmä som nezasiahla nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomá následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autorky

POĎAKOVANIE

Veľmi rada by som sa chcela poďakovať vedúcemu bakalárskej práce panu Ing. Adriánu Tomašovi za odborné vedenie, konzultácie, trpezlivosť, správne usmernenie a rady k práci.

Obsah

Zoznam obrázkov

Zoznam výpisov

Úvod

OS Linux predstavuje svetovo najrozšírenejší open source operačný systém. Tak je tomu práve preto, pretože podlieha zmenám, vylepšeniam, rozšíreniam každým momentom. Za tieto zmeny sú zodpovedné nespočetné skupiny vývojárov, ktorí neustále prispievajú svojimi dielami do distribúcií Linuxu. Tieto vylepšenia ale nie sú vždy dokonalé. Vznikajú zraniteľnosti, ktoré sú kritické pre správny a najmä bezpečný chod systému. Chyby môžme nájsť už v jadre operačného systému, ktorý je považovaný za dôverihodnú časť výpočetnej základni počítačových systémov. Je nevyhnutnou súčasťou komunikácie medzi užívateľom a hardvérom.

Ako sa Linux časom vyvíjal, pribúdali nové verzie, nové programy či aplikácie a spolu s nimi pribúdali aj zraniteľnosti, ktoré znižovali úroveň bezpečnosti systému. Vďaka svojej otvorenosti ale otvárali možnosti verejnosti tieto chyby nájsť a vydať na ne opravy. Otvorenosť ale otvára dvere aj útočníkom. Je potrebné teda svoj operačný systém pravidelne aktualizovať a využívať toho, že sa daná zraniteľnosť nedá napadnúť na vašom zariadení.

Obsah tejto bakalárskej práce spočíva v predstavení konkrétneho operačného systému, rôznych typov potencionálnych útokov a zraniteľností, ktoré napokon na daný systém využijeme a napokon sa aj nasadia aktívne ochranné prostriedky zabraňujúce týmto útokom. V teoretickej časti v kapitole 1 sa teda zameriame na popis OS, nášho konkrétneho jadra a jeho využitia. Vysvetlíme aj princíp UNIX, GNU/Linux a aj jeho distribúcie. Kapitola 2 sa zaoberá prehľadom aktívnych a pasívnych útokov na OS. Kapitola 3 popisuje zraniteľnosti, ktoré vie pokročilejší užívateľ odstrániť samostatne. Nasledujúca kapitola 4 obsahuje opis a výsledky skenovacích programov, ktoré využijeme ako pomôcku pre hlbšiu detekciu problémov. Kapitola 5 je venovaná výsledkom útokov zrealizovaných na základe zraniteľností, ktoré sme v predchádzajúcich krokoch našli a čo sme s nimi dosiahli. Kapitola 6 sa zaoberá aplikáciou ochranných prostriedkov proti vykonaným útokom a ich opisom. Posledná kapitola 7 pojednáva o komplikáciach, ktoré nastali pri ďalších riešeniach zraniteľností.

1 Základný koncept Linuxu

Obsahom tejto kapitoly je teoretický popis operačného systému Linux. Zahŕňa témy týkajúce sa operačného systému, jeho jadra - kernelu, taktiež sú opísané začiatky Unixových systémov, ozrejmí sa vznik celkového systému GNU/Linux a upresní sa aj distribúcia na ktorej naše Linuxové zariadenie beží.

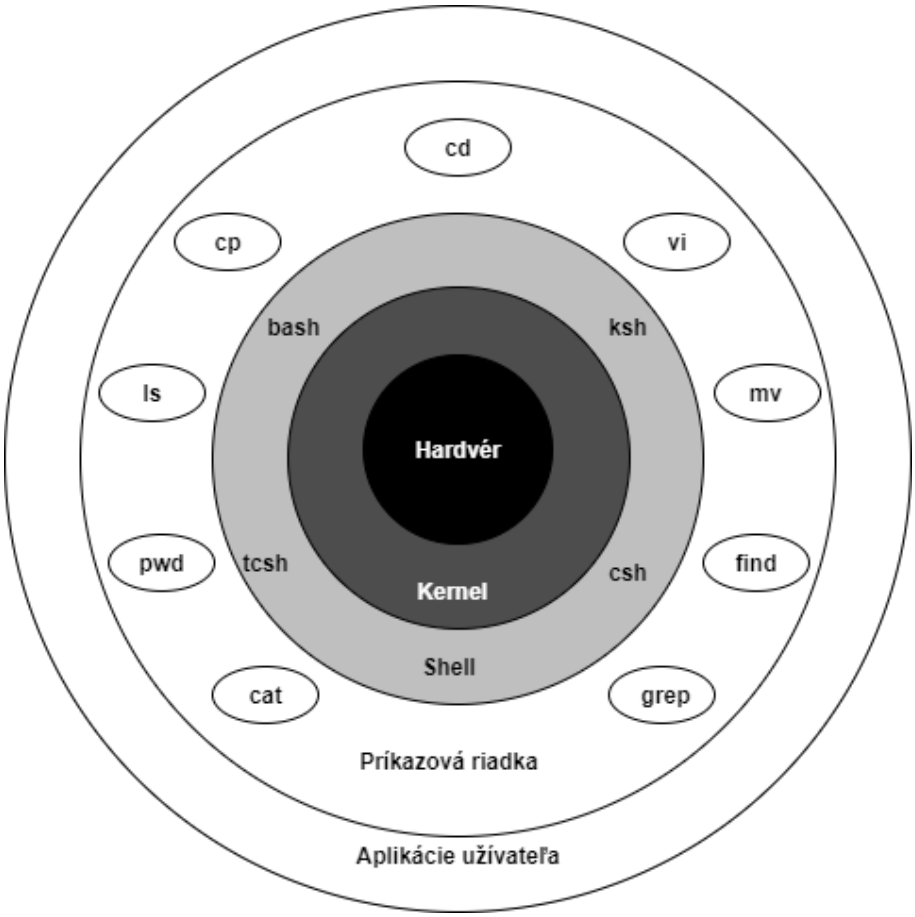
1.1 Operačný systém

Operačný systém, známy pod označením OS, je softvér, ktorý je základom pre komunikáciu užívateľa počítača s hardvérom. Jeho účel pozostáva z poskytnutia vhodného prostredia, v ktorom je užívateľ schopný spúšťať a ovládať aplikačné programy pohodlne a efektívne. Kvôli tomu OS manažuje hardvér tak, aby zaistil mechanizmy zaisťujúce správne operácie počítačového systému a prevenciu od rušivých operácií. Tvorí základnú množinu softvéru, ktorý manipuluje so vstupnými (klávesnica, myš) a výstupnými (tlačiareň, obrazovka) zariadeniami pomocou ovládačov týchto zariadení vyvinutými tvorcami hardvéru. Ďalej sa stará o systém ukladania súborov pre možné manipulovanie s dátovými súbormi v aplikáciach a tiež spravovať súbory, ktoré spúšťajú aplikácie. Pritom musí efektívne alokovať zdroje ako pamäť, procesor a úložisko. Pri užívaní OS je potrebné využiť všetky jeho funkcie a tým znížiť potenciál útokov smerovaných na systém. Dosiahnuť to môžeme napríklad pravidelnými aktualizáciami systému, inštalovaním antivírusového softvéru, konfiguráciou firewalu – filtrovaním premávky, či obmedzením práv užívateľov systému [1].

1.2 Jadro – Kernel

Jadro systému – kernel – je najzákladnejšou vrstvou operačného systému. Tvorí elementárny základ operačného systému a koordinuje všetky funkcie počítača. Pomocou medziprocesovej komunikácie a systémových volaní ho môžeme chápať ako most medzi aplikáciami a spracovávaním dát vykonávaných na hardvérovej úrovni. Pri načítaní OS v pamäti sa kernel načítava ako prvý a ostáva aktívny až dokiaľ sa znovu OS nevypne. Vykonáva funkcie ako správu procesoru, správu pamäti a správu vstupných a výstupných zariadení. Zdrojový kód kernelu je uložený v chránenej časti pamäti kvôli zabezpečeniu pred prepísaním iným programom.

Prostredníctvom CPU pracuje v privilegovanom režime, čo znamená, že má plný prístup ku hardvéru a zamedzuje priamym interakciám užívateľských programov s týmto hardvérom [2].



Obr. 1.1: Jadro vzhľadom na ostatný systém

1.3 UNIX

UNIX predstavuje operačný systém koncipovaný ako multiužívateľský a multiprosecový. Bol vyvinutý v laboratóriu spoločnosti Bell (spadajúca pod AT&T) a neskôr prepísaný do jazyka C (D. Ritchie), čím autori zaistili ľahšej prenosnosti programov. Užívatelia k OS UNIX pristupujú cez terminál – ich práva sú jednoznačne vymedzené, čo zachováva maximálnu ochranu pred neodbornými zásahmi. Patrí k otvoreným systémom, čo znamená, že splňuje kritériá možnosti interoperability, portability a otvorených štandardov.

UNIX sa stal predlohou pre vývoj mnohých OS ako napríklad : GNU/Linux, macOS, Darwin, AIX, či Solaris. Je populárny aj vďaka nasadeniu UNIX-ovej filozófii, ktorú založili Ken Thomson spolu s ďalšími developérmi, aby zabezpečili jedinečné vlastnosti [3].

Filozofia pozostáva z týchto pravidiel:

1. *Small is beautiful* – menšie časti sa spájajú lepšie ako väčšie a komplexnejšie
2. *Make each program do one thing well* - sústredenie na jednu úlohu predchádza množstvu nadbytočného kódu, ktorý často spôsobuje prehrievanie, úbytok flexibility a nežiadanú komplexitu
3. *Build a prototype as soon as possible* – prototyp sa berie ako základ pre ďalší vývin efktívneho dizajnu
4. *Choose portability over efficiency* – toto pravidlo sa stalo základom aj pre iné systému okrem UNIX-u
5. *Store numerical data in flat ASCII files* – kvôli prenosnosti
6. *Use software leverage to your advantage* – opakované používanie modulov kódu pre rýchly development
7. *Use shell scripts to increase leverage and portability* – uprednostnenie písania skriptov než programov v jazyku C
8. *Avoid captive user interfacces* – vyhýbanie sa vytváraniu príkazov, ktoré užívateľa obmedzujú a nedovoľujú mu na pozadí iného príkazu vykonávať ďalší
9. *Make every program a filter* – tvoriť programy tak, aby vedeli dáta vytvárať a nie len ich modifikovať [4].

1.4 GNU/Linux

GNU označuje systém vyvinutý v rámci projektu GNU. V tomto projekte sa Richard Stallman zaoberal vývinom kompletne slobodného operačného systému inšpirovanému UNIX-om. Popri vývoji tohto OS nastali komplikácie, ktoré im nakoniec vyriešil Linus Torvalds, ktorý sa rozhodol zverejniť zdrojové kódy svojho jadra – Linux. Spojením GNU operačného systému s Linuxom vznikol dnes hojne používaný OS – GNU/Linux. Stallman vytvoril filozofiu o GNU/Linux OS, ktorá hovorí o jeho cieli udržať OS úplne slobodný od iných neslobodných (komerčných) komponent.

GNU/Linux uskutočnil prevrat v otvorenosti celého systému verejnosti bezplatne a ponúka slobodné prostredie pre ďalší vývin, distribúciu či pre akademické účely. Aj v dnešnej dobre si drží svoju popularitu v podiele operačných systémov používaných vo svete, síce nevedie v platformách určených pre desktopových užívateľov ale zato vyniká v zastúpení webových serverov a superpočítačoch [5].

1.5 Distribúcie – Debian

Linux distribúciu tvorí množina komponent a procesov, ktoré sú požadované pre dosiahnutie správne fungujúceho systému. Dnešné Linuxové distribúcie obsahujú

kernel, GNU shell progamov (terminál a príkazy), zobrazovací systém (X Window), desktop prostredie (GUI - GNOME) či systém riadenia balíčkov. Veľkou výhodou je, že všetky tieto komponenty je možné vymieňať za iné podľa potreby užívateľa alebo možností hardvéru. Napríklad pre potreby serveru je potrebné mať systém čo najmenší a najvýkonnejší. Naopak distribúcie určené pre osobné počítače zväčša GUI obsahujú a nekladú sa na ne také vysoké požiadavky. Odhaduje sa, že existuje okolo 600 Linuxových distribúcií, z ktorých je asi 500 v aktívnom vývine.

Medzi tie najpopulárnejšie patria:

- Debian
- Fedora
- Ubuntu
- CentOS
- Kali Linux
- Linux Mint
- Red Hat Enterprise

Systém, ktorý je predmetom tejto práce je distribúcia Debian. Už pri jednoduchej inštalácii obsahuje okolo 59 000 balíčkov. Je lídrom vzhľadom na úroveň jeho vývinu a od ostatných distribúcií sa odlišuje svojím systémom manažovania balíčkov. Tento nástroj dáva administrátorovi systému úplnú kontrolu nad správou týchto balíčkov – jednotlivé/automatické aktualizovanie, príkazy pre zachovanie určitej verzii balíčku či označenie softvéru pridaného z neoficiálnych zdrojov spolu s jeho funkciami [6].

2 Typy útokov na Linux OS

Existujú rôzne typy útokov, hrozieb a zraniteľností, ktoré poškodzujú a narúšajú systémovú bezpečnosť. Tieto útoky sa rozdeľujú na dva typy a to aktívne a pasívne. Rozdiel medzi týmito typmi je, že aktívnymi útokmi sa útočník snaží zachytiť spojenie a modifikovať dáta/informácie. Cieľom pasívneho útoku je zachytenie premávky a informácií o nej s úmyslom ich analyzovať a vyťažiť z nich všetky potrebné informácie.

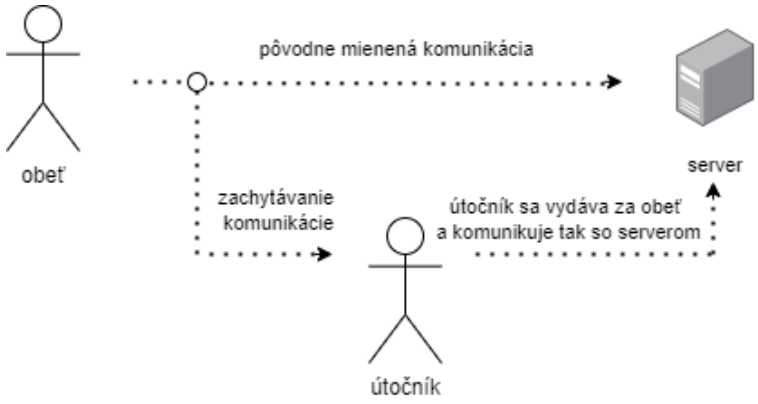
2.1 Aktívne útoky

Pri aktívnych útokoch narušiteľ zasahuje do komunikácie medzi zariadeniami. Dáta, ktoré si komunikačné zariadenia posielajú medzi sebou dokáže sledovať a dostať k nim priamy prístup. Bez toho, aby sa niečo obe strany dozvedeli dokáže zmeniť prenášané dáta, vyvoláva šum pri ich prenášaní a vkladá chybné bity do prenosu. Týmito akciami útočník oslabuje integritu a dostupnosť – dôležité zložky informačnej bezpečnosti.

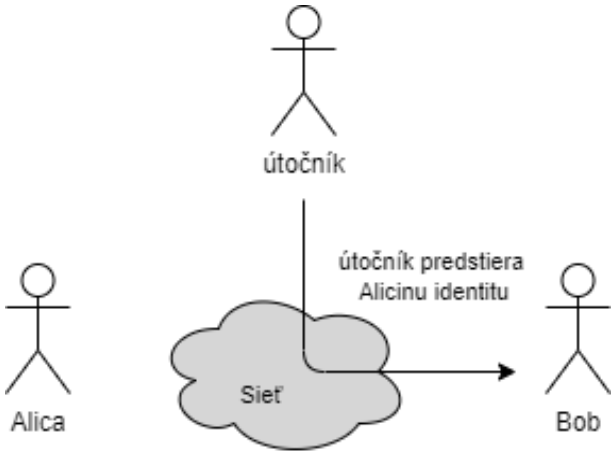
Integrita dát znamená že sú behom ich životného cyklu zabezpečené proti modifikácii či vymazaní neautorizovanou stranou a taktiež zaisťuje ochranu pred zmenami, ktoré by mohli poškodiť dáta a systém. Dostupnosť zasa zabraňuje narušeniu dostupnosti služieb systému ako autentifikačné mechanizmy či prístupové kanály.

Typy aktívnych útokov:

1. *Replay* – nastáva v momente, keď neautorizovaný užívateľ zachytí sieťovú komunikáciu a túto komunikáciu pošle na jej originálny cieľ popritom správajúci sa ako pôvodný odosielateľ. Tomuto typu útoku sa dá zabrániť implementovaním časových známok a sekvenčných čísiel na posielané pakety, čo umožňuje autentifikačnému systému prijať iba tie pakety, ktoré tieto čas. známky a sekvenčné čísla obsahujú. Ak sa napríklad časová známka nachádza v neprimeranom rozmedzí, paket je zahodený. Príkladom replay útoku je *session replay* – jedinečné relačné ID užívateľa je ukradnuté útočníkom pri odpočúvaní komunikácie medzi obeťou a serverom. Táto komunikácia sa po zachytení znovu identicky odošle na server. Pod týmto ID je možné sa vydávať za poškodeného autorizovaného užívateľa a získaným prístupom využívať jeho práva. [7].
2. *Masquerade* – maškaráda predstavuje útok, ktorý využíva falošnej identity ku získaniu neautorizovaného prístupu k informáciam prostredníctvom legitímnej prístupovej identifikácii (využitie napr. replay útoku). Detekcia typu tohto útoku je podmienené neobvyklým správaním užívateľa [8].



Obr. 2.1: Replay attack



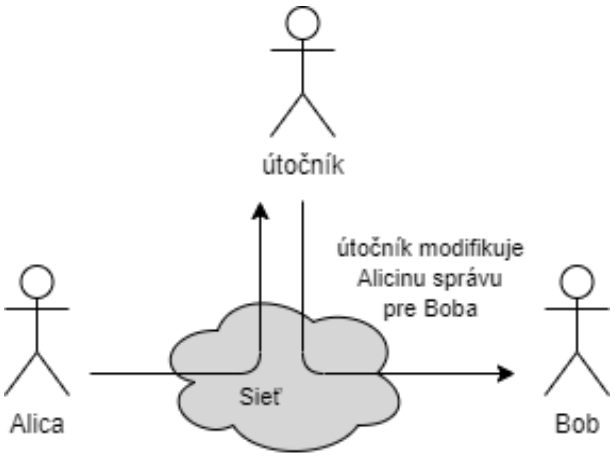
Obr. 2.2: Masquerade

3. *Modifikácia obsahu komunikácie* – znamená úpravu časti legitímnej správy alebo jej oneskorené či poprehadzované doručenie, čo evokuje neautorizovaný zásah (obr. 2.3) [9].
4. *Denial of Service* – spočíva v zaplavení premávky žiadosťami o nadviazanie komunikácie na strane serveru. Žiadosti sú nelegitímne a ich zdrojová adresa je vymyslená aby na ne server nedokázal odpovedať. Týmto je server pohltený a padá (obr. 2.4).
   Príklady DoS útokov :

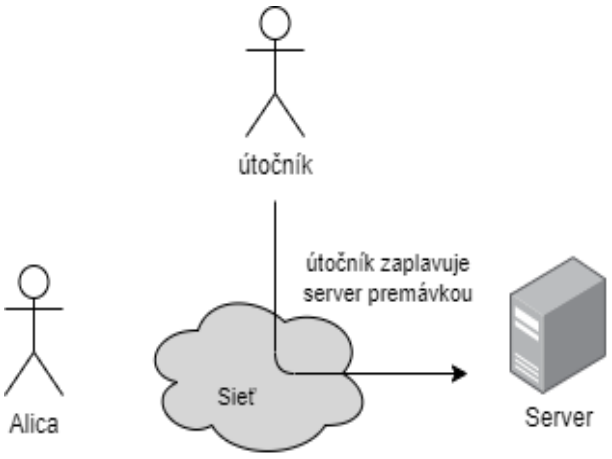
   (a) *smurf attack* – útočník rozosiela prostredníctvom broadcastu Internet Control Message Protocol (ICMP) pakety so zdrojovou IP adresou obeti po sieti rôznym hostiteľom. Títo hostitelia reagujú na prijaté pakety a posielajú svoju odpoveď na IP adresu obeti, ktorá ale v skutočnosti žiadne žiadosti neodosielala. Obeť je teda pohltená odpoveďami a spomaľuje sa jej celý systém.

(b) *SYN flood* – na cielený server útočník zasiela požiadavok na zahájenie komunikácie prostredníctvom *three-way handshake* metódy využívanej v protokole TCP/IP. Server naň odpovie, ale dokončenie handshaku útočník neuskutoční a tým pádom necháva port, na ktorom komunikovali okupovaný – nemožno s ním ďalej komunikovať. Takto sa okupujú aj ostatné voľné porty až legitímnym užívateľom neostanú žiadne prítupné.

Existuje ešte vylepšená verzia DoS a to *Distributed DoS*, ktorá na svoje útoky využíva väčšie množstvo zariadení pracujúcich proti jednej obeti. Tieto zariadenia sa po útoku hackera stávajú súčasťou botnetu – sieti robotov ovládaných hackerom [10].



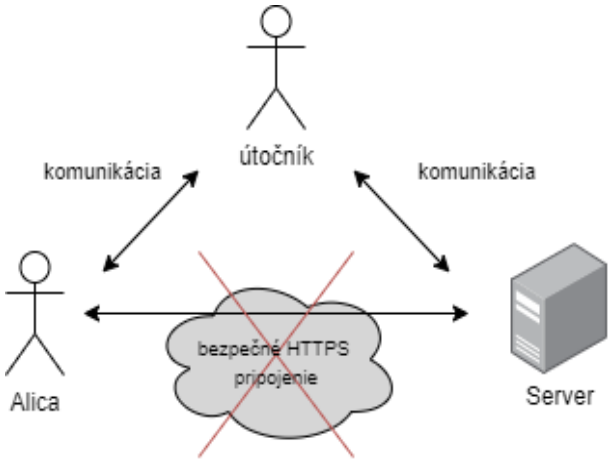
Obr. 2.3: Modifikácia obsahu komunikácie



Obr. 2.4: Denial of service

5. *Man in the Middle* – pozostáva z nasadenia narušiteľa do komunikácie medzi 2 strany kvôli sledovaniu alebo manipulácii s premávkou. Útočník zasahuje

do legitímnych sietí alebo si vytvára svoju falošnú, ktorú celkovo kontroluje. Premávka je napokon zbavená šifrovacích algoritmov pre ľahšiu manipuláciu. Po využití informácií, útočník znovu využitú časť premávky zašifruje aby nebolo poznať, že bola zneužitá [11].



Obr. 2.5: Man in the Middle

2.2 Pasívne útoky

Pasívne útoky sa zameriavajú na pozorovanie, monitorovanie a kopírovanie informácií. Útočník sa zameriava na otvorené porty a rozličné sieťové zraniteľnosti aby si dokázal zaistiť prístup do systému. Pri tomto type útokov sa kladie dôraz na prevenciu, napríklad nasadenie silných kryptografických metód, ktoré zabezpečia riadne šifrovanie správ premávajúcich po sieti alebo konfigurácia firewalu.

Narozdiel od aktívnych útokov, pasívne sú pre obeť nepostrehnuteľné. Aj preto je dôležitá dôkladne zabezpečená komunikácia, ktorá zabráni premene pasívneho útoku na aktívny, ktorý je nebezpečnejší. Pasívne útoky taktiež porušujú zložku informačnej bezpečnosti – CIA triády – a to dôvernosť. Dôvernosť zaručuje prístup k citlivým informáciám len autorizovaným osobám. Implementuje sa mechanizmami ako užívateľské mená, heslá, ACL zoznamami a šifrovaním.

Príklady pasívnych útokov:

1. *Odpočúvanie* – útočník načúva a snaží sa zachytiť posielané správy, ktoré prinajlepšom nie sú šifrované (napr. zaslanie hesla ako odpoveď pri http žiadosti)
2. *Analýza premávky* – sledujú sa prenášané metadáta z ktorých útočník odvodí čo najviac informácií o priebehu výmeny a o účastníkoch a to aj v prípade šifrovaného prenosu [12].

3 Potenciálne zraniteľné miesta v systéme

Pri obstarávaní Linux OS je dôležité už pri prvotnom užívaní implementovať dobré security pravidlá. Linux ako taký pri prvom spustení neobsahuje dostatočne zabezpečený systém – je nutné si uvedomiť, na čo sa tento systém bude využívať a podľa toho vybrať vhodné nastavenia. Táto kapitola sa teda zaoberá skúmaním konfigurácie systému – čo sa dá vylepšiť a čo sa dá napadnúť.

3.1 Heslá a užívateľské účty

Jedna z prvých vecí, na ktorú užívateľ narazí pri spustení systému je prihlasovacie meno a jeho heslo. Heslo ako také, nezáleží kde je použité, by malo byť unikátne – to znamená, že by ho užívateľ nemal používať 2 krát. Všeobecne sa naň dá aplikovať zopár pravidiel, ako zabrániť jeho kompromitovaniu.

Tieto kritériá zahŕňajú:

- vytvárať heslá s dĺžkou aspoň 8 znakov
- znaky v hesle by mali obsahovať veľké a malé písmená, čísla a symboly
- heslo by nemalo obsahovať slovníkové výrazy, frázy, či osobné informácie užívateľa
- ak nieje systémovo nastavené, tak by mal užívateľ pravidelne meniť svoje heslá
- všeobecne ale platí, čím je heslo zložitejšie, tým lepšie [13].

Ďalšie kritériá zabezpečuje administrátor systému s užívateľským právom `root`. Nastaví napríklad životnosť hesiel – názory na jej dĺžku sú rôzne ale sú v rozmedzí 30 až 90 dní. Administrátor zaistí, aby bol pri končení tejto periódy užívateľovi zaslaný email s upozornením o vypršaní platnosti hesla. Taktiež administrátor nastaví, aby bol užívateľ po určitej časovej perióde nečinnosti odhlásený a teda prinútený sa prihlásiť znovu.

Možné nebezpečenstvo vzniká aj pri skutočnosti, kedy užívateľské heslá možno vyňať zo systému počas bootovania pri priamom fyzickom prístupe a bez predošlej autorizácie. Prostredníctvom `recovery` módu sa útočník dostane do `root shell prompt` príkazovej riadky, z ktorej zmení práva `root` z `read only` na `read write`, ktoré mu umožnia zmeniť heslo `root` alebo akéhokoľvek iného užívateľa.

Užívateľské účty by taktiež mali spadať pod určitú úroveň obmedzenia. Administrátor zaistí, aby užívateľovi nebol odopretý prístup tak, aby nemohol vykonávať svoju prácu. Naopak užívateľa musí obmedziť v určitom rozmedzí aby nebol schopný zasahovať do vecí, ktoré by mohol modifikovať a poškodiť. Taktiež zaistí, aby mal užívateľ presne toľko privilégií koľko je potrebných pri vykonávaní jeho práce.

Pri väčších systémoch napríklad v korporáte je dôležité, aby mali všetci administrátori správne rozdelené práva a mali prístup do systému cez `sudo` funkciu než cez `root`, ktorá je príliš rozsiahla. Hlavný administrátor má potom lepší prehľad o tom, ako sa so `sudo` právami narába [14].

3.2 Secure Shell

Secure Shell (SSH) predstavuje nástroj pre bezpečný vzdialený transfer súborov a pre vzdialené prihlásenie do systému. Po inicializačnej negociácii je prenos šifrovaný dohodnutou šifrou (napr. AES). Samotné SSH so základnou konfiguráciou nieje až tak bezpečné, preto je potrebné ho posilniť. Jeho konfigurácia je uložená v `/etc/ssh/ssh_config`.

Rozbor konfigurácie:

- *verzia protokolu* – existujú 2 verzie, SSH version 2 je považovaná za najbezpečnejšiu a ak nieje inak potrebná tak verzia 1 by mala byť zakázaná,
- *SSH kľúč* – tento kľúč by mal byť generovaný algoritmom RSA s aspoň 3072 bitovým kľúčom alebo Elliptic Curve Digital Signature Algorithm (ECDSA) s 384 bitovým kľúčom. Takto silné šifrovacie algoritmy nás pripravujú na mimoriadne silnú kvantovú výpočtovú techniku. Vygenerovaný verejný kľúč je potom prenesený na server a teda nieje potrebné prenášať heslo cez sieť. Namiesto hesla sa teda užívateľ prihlasuje prostredníctvom prístupovej frázy jeho súkromného kľúča, ktorú si nastavil,
- *zákaz root prihlasovania* – v konfiguračnom súbore je podstatné zakázať prihlasovanie pomocou `root` užívateľa
- *port* – štandartne je nastavený port číslo 22. Pre sťaženie práce útočníkovi je dobré toto číslo zmeniť na iný voľný port,
- *predvolené šifrovacie algoritmy* – ich zoznam sa nachádza v už zmienenom konfiguračnom súbore. Tie algoritmy, ktoré sú považované za slabé teda vyhodíme,
- *logovanie do SSH* – pre extra mieru bezpečnosti sa aplikuje mechanizmus kontroly prístupu pre zredukovanie užívateľov a skupín užívateľov, ktorí by mohli SSH využívať (DenyUsers, AllowUsers, DenyGruops, AllowGroups),
- *automatické odlhásenie* – po určitej časovej perióde je užívateľ automaticky odhlásený [15].

3.3 Update

Aktualizovanie Linux systému je obecne pohodlnejšie než napríklad OS Windows. Ku klasickému aktualizovaniu postačí zadať dva príkazy. Taktiež pre odstránenie nepotrebných balíčkov je možné použiť jednoduchý príkaz. Dá sa využiť aj automatické aktualizovanie pomocou `cron-apt` – ten ale vie balíčky len sťahovať a aktualizovať ich zoznam. Nevýhoda je, že ich nedokáže nainštalovať. Taktiež sa neodporúča tento nástroj používať v systémoch, ktoré sú nestabilné alebo v stave testovania.

3.4 Odstránenie zbytočných konfiguračných súborov

Dôkladnejšia údržba a odstraňovanie nepotrebného softvéru v systému je esenciálne pri udržovaní dostatočného miesta na disku najmä pri tých s malou kapacitou. Všeobecne, ak je to možné, je lepšie udržiavať menší počet balíčkov v systému vzhľadom na potenciálne hrozby. Pomocou rozličných príkazov sa dá odstrániť celý balíček, jeho závislé zložky, konfiguračné súbory či uložené archívy balíčkov, ktoré sa už nedajú stiahnuť. Takto sa systém dokáže zbaviť napríklad súborov, ktoré systém nevyužíva – siroty, návodov v cudzích jazykoch, súborov spojených s ipv6 (ak ju teda nevyužívame) alebo dokáže vymeniť stávajúce balíčky za ich menšie ekvivalenty.

3.5 Bootovanie externých zariadení

Pre manažment pripojených zariadení v Linuxe sa využíva nástroj `udev`. Prostredníctvom neho systém manažuje a detekuje zariadenia pripojené do systému. Dynamicky vytvára a odstraňuje uzly, ktoré predstavujú rozhranie pre ovládač pri bootovaní alebo pridávaní a odstraňovaní zariadenia zo systému. Šíri informácie o týchto zariadeniach alebo o zmenách ich stavu do užívateľského priestoru. Jeho ďalšie funkcie pozostávajú zo zásobovania systému eventami zariadenia, spravovania povolení uzlov, či vytvárania pravidiel. Tieto pravidlá majú najvyššiu prioritu a užívateľ si ich môže prispôsobiť podľa potrieb. Taktiež predstavujú prekážku pre zariadenia, ktoré by nemali mať prístup do systému a vylúčiť potenciálnu bezpečnostnú hrozbu.

3.6 Otvorené porty a poskytované služby

Administrátor si musí uvedomiť, ktoré ponúkané služby využíva alebo by potenciálne v budúcnosti využiť mohol. Taktiež by mal mať prehľad nad službami, ktoré sú zastaralé a automaticky ich vypnúť. Zoznam služieb poskytovaných v systéme nájdeme v zložke `/etc/init.d`. To, či sú aktívne sa dá zistiť pomocou `service`

`--status-all`, kde sa podľa znamienok mínus a plus pozná ich stav. Zrušením nepotrebných služieb sa nielen prispeje ku vyššej výkonnosti systému ale samozrejme sa prispeje k menšej miere zraniteľnosti systému [16].

3.7 Log file

Log súbory obsahujú záznamy udalostí odohraných v systéme. Sú to veľmi dôležité a cenné informácie pre administrátora, ktoré mu umožňujú udržať prehľad nad významnými udalosťami. Ukrývajú sa v repozitári adresára `/var/log`.

Medzi najkritickejšie patria:

- `/var/log/messages`
- `/var/log/auth.log`
- `/var/log/secure`
- `/var/log/boot.log`
- `/var/log/dmesg`
- `/var/log/kern.log`
- `/var/log/faillog`
- `/var/log/cron`
- `/var/log/yum.log`
- `/var/log/httpd/`

Pri pribúdajúcich záznamoch je obtiažne udržiavať rozhľad, preto je využitie pomocných nástrojov na mieste. K tomu slúži napríklad nástroj *Logcheck*, ktorý napomáha administrátorovi s analýzou a skenovaním log súborov [17].

3.8 Zálohovanie

Jedna z vecí, ktorá umožní administrátorovi systému pokojne spať je pravidelné zálohovanie systému. Náhoda môže zapríčiniť nepríjemné situácie ako stratu/poškodenie dát, poškodenie počítača a mnoho ďalších nočných moriem. Preto je namieste vykonávať zálohu primerane často vzhľadom na množstvo uskutočňovaných zmien. Záloha sa vykonáva rôznymi spôsobmi podľa potrieb. Zálohuje sa kompletný systém, určité dáta, konfigurácie balíčkov či rôzne kombinácie. Existujú napríklad kompletné zálohy, inkrementované, reverzné či neštruktúrované. Užívateľ si spomedzi nich vyberá a uplatňuje takú, ktorá mu zaistí čo najefektívnejšie zotavenie systému [18].

4 Využitie skenovacích programov na hlbšiu detekciu

V každom systéme či aplikácii sa nachádzajú nedokonalosti, ktoré sa transformujú do nechcených bezpečnostných zraniteľností. Samozrejme, ani Linux nieje perfektný, preto je dôležité poznať čo najdetailnejšiu konfiguráciu systému a pravidelne sa o systém starať. Najlepšie je odhaliť zraniteľnosti skôr, než to dokáže útočník. S touto prácou pomôžu rôzne druhy skenovacích nástrojov poskytujúce informácie opisujúce stav systému a jeho častí.

4.1 Lynis

Predstavuje bezpečnostný nástroj určený pre systémy s OS Linux, macOS – teda na bázi Unix. Vykonáva rozsiahlu kontrolu, ktorá po analyzovaní pomáha nachádzať nedostatky v zabezpečení. Je to skenovací, modulárny a oportunistický nástroj – pre jeho plnohodnotný chod nieje nutná inštalácia iných prídavných zariadení, čo udržiava systém od ďalšieho softvéru. Pri skenovaní vždy hľadá a rozširuje zdroje, čo ponúka rozsiahly report. Taktiež umožňuje vykonávať testy pripravené individuálne aj v iných skriptovacích či programovacích jazykoch a keďže je modulárny, prijíma externé rozšírenia s ktorými sa report prispôsobí potrebám užívateľa. Výstup skenu sa rozdeľuje na sekcie a podsekcie označujúce potenciálne zraniteľnosti podľa služieb alebo logických skupín označené bezpečnostným stavom `[OK]` – značí dobrý výsledok, čo ale neznamená, že je daná komponenta dostatočne nakonfigurovaná. Ďalším najčastejším stavom je `[WARNING]` – evokuje upozornenie, sken narazil na niečo neočakávané čo ale automaticky neznamená, že je to chybné. Je potrebné upriamiť pozornosť na komponentu s týmto stavom a ubezpečiť sa, že je všetko v poriadku. Ak konfigurácia vyhovuje, je možné do budúcna vynechať sken tejto komponenty, aby naďalej zbytočne nemiatla. Výsledky sa ukladajú do zložky `/var/log/lynis.log` a `/var/log/lynis-report.dat`, kde sa nachádzajú návrhy na odstránenie problému alebo poskytujú detailnejšie informácie o výsledku skenu [19].

Ukážka skenu na výpise 4.1 vyobrazuje inicializáciu procesu. Je z neho poznať aktuálne používané verziu, základné charakteristiky systému ako operačný systém, jeho verzia, verzia jadra, hardvérová platforma, na ktorej beží. Ďalej vyobrazuje dôležité adresy výsledkov skenov a meno užívateľa, ktorý vykonáva aktuálny sken.

Výpis 4.1: Proces inicializácie skenu

```
[ + ] Initializing program

-- Detecting OS...                                     [ DONE ]
-- Clearing log file (/var/log/lynis.log)...           [ DONE ]

Program version:           1.6.3
Operating system:          Linux
Operating system name:     Debian
Operating system version:  8.9
Kernel version:            3.18.12
Hardware platform:         i686
Virtual machine:           Unknown
Hostname:                  localhost
Auditor:                   root
Profile:                   /etc/lynis/default.prf
Log profile:               /var/log/lynis.log
Report file:               /var/log/lynis-report.dat
Report version:            1.0
Plugin directory:          /etc/lynis/pluging
```

4.1.1 Lynis – ukážka skenu

Jedna z mnohých sekcií skenovaných komponent je aj využívaný Shell – predstavuje rozhranie medzi užívateľom a systémom. Príkazy zadané užívateľom cez príkazovú riadku berie ako vstup a ten sa snaží vykonať. Existujú dva typy shell rozhraní, a to: *Bourne shell* a *C shell*.

Na aktuálnom zariadení z výpisu 4.2 môžeme poznať, že sa v systéme nachádza päť druhov shellov v zložke `/etc/shells`. Po bližšom skúmaní zložky tvorí skupinu shellov `sh`, `dash`, `bash`, `rbash` a pre ovládanie okien `screen`. Využívaný shell na tomto zariadení je typu bash (Bourne Again shell) verzie 4. 3. 30 (1). Ako konfiguráciu, ktorá by sa dala vylepšiť sken zaradil vlastnosť `Session timeout` – ktorá zabezpečuje vypnutie relácie užívateľa po určitej časovej perióde pasívneho chovania. Inak sken vyhodnotil vybrané zraniteľnosti zo zoznamu *CVE* ako neškodné.

Výpis 4.2: Shell sken

```
[ + ] Shells

-- Checking shells from /etc/shells
   Result: found 5 shells (valid shells: 5)
   -- Session timeout settings/tools                    [ NONE ]

-- Testing for Shellshock vulnerability                 [ OK ]
   -- CVE-2014-6271 (original shellshocker)             [ OK ]
   -- CVE-2014-6271 (segfault, lcamtuf bug #1)          [ OK ]
   -- CVE-2014-6278 (Florians patch, lcamtuf bug #2)    [ OK ]
   -- CVE-2014-7169 (taviso bug)                        [ OK ]
   -- CVE-2014-7186 redir_stack bug                     [ OK ]
   -- CVE-2014-7187 nested loops off by one bug         [ OK ]
   -- Expliot#3 on sheellshocker.net (no CVE)           [ OK ]
```

Pri kontrolovaní SSH komponenty na výpise 4.3 Lynis upozorňuje na možnosť `PermitRootLogin`, teda možnosť pripojiť sa do SSH relácie pomocou účtu `root`, čo je považované za potenciálne nebezpečenstvo – preto by mala byť táto možnosť zakázaná. Na výpise 4.4 je upozornenie vyobrazené v `lynis-report.dat`.

Výpis 4.3: SSH sken

```
[ + ] SSH Support

-- Checking running SSH deamon                          [ FOUND ]
   -- Searching SSH configuration                       [ FOUND ]
   -- Checking defined SSH option                       [ DONE ]
   -- SSH option: PermitRootLogin                       [ WARNING ]
   -- SSH option: Protocol                              [ OK ]
   -- SSH option: StrictModes                           [ OK ]
   -- SSH option: AllowUsers                            [ NOT FOUND ]
   -- SSH option: AllowGroups                           [ NOT FOUND ]
```

Výpis 4.4: SSH report

```
Warning []=SSH-7412:Root can directly login via SSH!
```

4.2 Nmap

Inak známy aj ako *Network Mapper* už názvom naznačuje svoj účel – reprezentuje jeden z najrozšírenejších skenovacích nástrojov siete. Pomáha odhaľovať nedostatky a zraniteľnosti na základe identifikovania zariadení bežiacich v sieti, zisťovaní dostupných hostiteľov, hľadaní otvorených portov a ponúkaných služieb. Zvládne monitorovať aj rozsiahle siete obsahujúce tisíce zariadení a podsietí.

Princíp skenovania spočíva v rozosielaní *raw* paketov do systémových portov. Následne počúva odpovede a určuje, ktoré porty sú otvorené a ktoré zatvorené alebo či sú nejakým spôsobom filtrované (firewall). Odpovede pozostávajú z paketov a informácií, ktoré nesú.

Ako jedna z užitočných nadstavieb je grafické užívateľské prostredie *Zenmap* poskytujúce mnoho rozširujúcich možností ako ukladanie skenov do databáze a ich porovnávanie, zobrazenie topológie sieti, či prezeranie aktívnych portov zariadení.

4.2.1 Ukážka skenu Nmap

IP adresa zariadenia prešla skenom a na výpise 4.5 je vidieť jej report pozostávajúci z hodnoty oneskorenia a počtu neaktívnych portov. Report taktiež jedná o otvorených portoch a službách na nich bežiacich – predstavujú ich služba ssh na porte číslo 22 a služba https na porte 443.

Výpis 4.5: Nmap sken

```
Starting Nmap 7.91SVM ( https://nmap.org ) at 2020-12-08 13:08 CET
Nmap scna report for 192.168.100.10
Host is up (0.00000030s latency).
Not shown: 998 closed ports
Port         STATE       SERVICE
22/tcp       open        ssh
443/tcp      open        https

Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds.
```

5 Praktický výsledok útokov

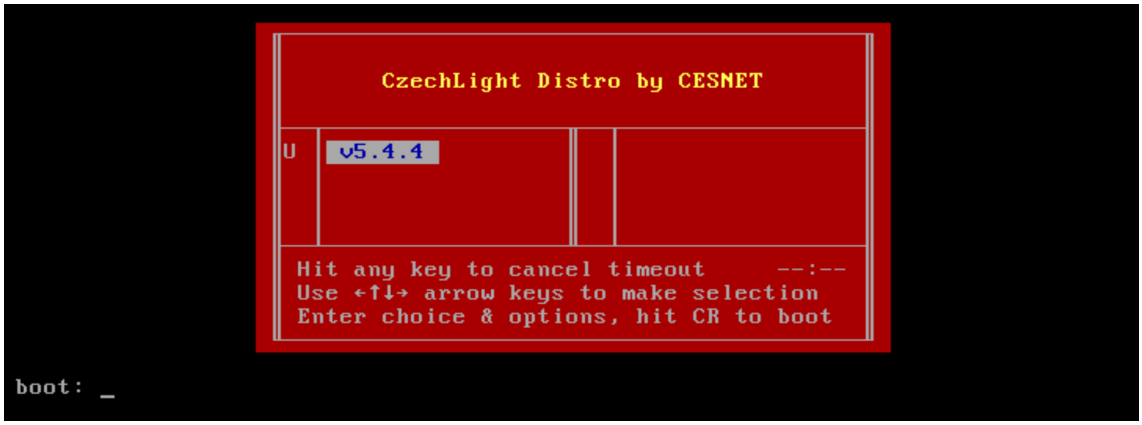
Praktická časť tejto práce sa bude v nasledujúcich kapitolách zaoberať aplikáciou rôznych útokov na zastaralý software. Zameria sa na oblasť problematiky hesiel a ich odhaľovanie pomocou brute force, ďalej veľmi častých útokov typu denial of service, vyskúša sa zraniteľnosť procesoru a po hlbšej detekcii pomocou skenovacieho softwaru sa útočí na bežiacu aplikáciu.

5.1 Password recovery útok

Pôvodne je password recovery určený pre užívateľa, ktorý zabudol/nemá prístupové heslo do systému. Napríklad u systémových administrátorov je pravdepodobné pri spravovaní väčšieho množstva systémov, že raz zabudnú niektoré z prístupových údajov a musia si vypomôcť obnovou hesla prostredníctvom bootovania systému.

Pre vykonanie tohto útoku je potrebný fyzický prístup k systému, ktorý je komplikovanejší dosiahnuť než útok po sieti. Taktiež s priamym fyzickým prístupom ku systému sa dajú napáchať oveľa väčšie škody, ale ak útočník nepozná heslo, môže začať s týmto.

Pri bootovaní zariadenia je potrebné stláčať klávesu *delete* aby sa útočník vedel dostať do menu bootovacieho manažéra. V našom prípade je na systéme nainštalovaný LILO manažér.



Obr. 5.1: Lilo boot manažér

Ako jediné spustiteľné zariadenie sa v menu ponúka linux `v5.4.4`. To spustíme v `single` móde pomocou príkazu `v5.4.4 init=/bin/bash`. Týmto útočník získal prístup do `root shell prompt`. Z príkazovej riadky sa zmenia prístupové práva z `ro (read-only)` predstavujúce právo len na čítanie na `rw (read-write)`, teda čítanie s možnosťou prepísania obsahu. Znázornenie je na výpise 5.1 .

Výpis 5.1: Zmena práv

```
root@(none):/# mount -n -o remount, rw /
root@(none):/#
```

Po úspešnom prepísaní práv má útočník voľnú ruku pri zmene hesla akéhokoľvek užívateľa. Stačí ak zmení heslo toho najdôležitejšieho – `root` (ak v systéme existuje-/existuje priamo pod menom `root`). K tomu postačí základný príkaz na výpise 5.2 `passwd` s upresnením, akého užívateľa si praje napadnúť.

Výpis 5.2: Zmena hesla

```
root@(none):/# passwd root
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated succesfully
root@(none):/#
```

Po úspešnej zmene hesla útočníkovi stačí vykonať reboot systému a prihlásť sa pomocou seba zadaného hesla ako je znázornené na výpise 5.3.

Výpis 5.3: Reboot

```
root@(none):/# exec /sbin/init
root@(none):/#
```

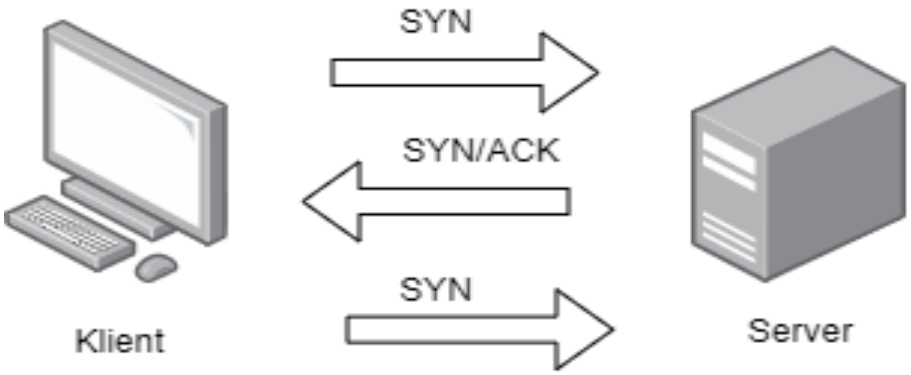
5.2 Denial of Service útok

Princíp tohto útoku je už popísaný v sekcii 2.1. V tomto type sa použila varianta *SYN flood*, čo znamená, že útočník po zistení voľného portu obeti započne pripojenie prostredníctvom TCP protokolu a posiela veľké množstvo SYN paketov bez úmyslu dokončenia riadneho nadviazania spojenia. Pre nadviazanie spojenia sa využíva *three-way handshake* na obr. 5.2, kedy klient pre započatie komunikácie pošle SYN paket, server pošle naspäť SYN/ACK paket a klient odpovedáom ACK paket – komunikácia je započatá.

Na tento útok je použitý nástroj `hping3` nainštalovaný na operačný systém Linux Ubuntu 20. 04. 1, z ktorého sa útočilo na náš zraniteľný systém s distribúciou Debian – obe v prostredí Oracle VirtualBox. Oba počítače sa nachádzali v rovnakej sieti, IP adresa obeti je 10. 0. 2. 6 a útočníka 10. 0. 2. 15. Ako sa už spomínalo, na útok sa využil nástroj `hping3` a cez príkazovú riadku stačilo zadať príkaz na výpise 5.4 a pomocou monitorovacieho softvéru – Wireshark – sa zachytávala posielaná komunikácia.

Výpis 5.4: hping3 príkaz pre započatie útoku

```
root@emilia-VirtualBox:# hping3 -S --flood -V -p 22 10.0.2.6
```



Obr. 5.2: TCP three-way handshake

Na výpise 5.4 je znázornený príkaz útočníka posielaný z príkazovej riadky z Ubuntu. Pozostáva z vyvolania programu `hping3`, nasleduje špecifikovanie typu paketov `-S` ako SYN pakety, `-flood` zabezpečuje rýchle posielanie paketov s ignorovaním odpovedí, `-V` slúži pre výpis informácií, `-p 22` označuje port, ktorý má systém zaplaviť spolu s dodanou IP adresou obeti.

No.	Time	Source	Destination	Protocol	Length	Info
91	0.002238315	10.0.2.15	10.0.2.6	TCP	54	55441 → 22 [RST] Seq=1 Win=0 Len=0
92	0.002266901	10.0.2.15	10.0.2.6	TCP	54	55443 → 22 [SYN] Seq=0 Win=512 Len=0
93	0.002277104	10.0.2.15	10.0.2.6	TCP	54	55444 → 22 [SYN] Seq=0 Win=512 Len=0
94	0.002285286	10.0.2.15	10.0.2.6	TCP	54	55445 → 22 [SYN] Seq=0 Win=512 Len=0
95	0.002294017	10.0.2.15	10.0.2.6	TCP	54	55446 → 22 [SYN] Seq=0 Win=512 Len=0
96	0.002305933	10.0.2.15	10.0.2.6	TCP	54	55447 → 22 [SYN] Seq=0 Win=512 Len=0
97	0.002358312	10.0.2.6	10.0.2.15	TCP	60	22 → 55442 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
98	0.002366437	10.0.2.15	10.0.2.6	TCP	54	55442 → 22 [RST] Seq=1 Win=0 Len=0
99	0.002399819	10.0.2.15	10.0.2.6	TCP	54	55448 → 22 [SYN] Seq=0 Win=512 Len=0
100	0.002451556	10.0.2.6	10.0.2.15	TCP	60	22 → 55443 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
101	0.002461227	10.0.2.15	10.0.2.6	TCP	54	55443 → 22 [RST] Seq=1 Win=0 Len=0
102	0.002478310	10.0.2.6	10.0.2.15	TCP	60	22 → 55444 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
103	0.002483479	10.0.2.15	10.0.2.6	TCP	54	55444 → 22 [RST] Seq=1 Win=0 Len=0
104	0.002496712	10.0.2.6	10.0.2.15	TCP	60	22 → 55445 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
105	0.002504690	10.0.2.15	10.0.2.6	TCP	54	55445 → 22 [RST] Seq=1 Win=0 Len=0
106	0.002516240	10.0.2.6	10.0.2.15	TCP	60	22 → 55446 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
107	0.002521513	10.0.2.15	10.0.2.6	TCP	54	55446 → 22 [RST] Seq=1 Win=0 Len=0
108	0.002534337	10.0.2.6	10.0.2.15	TCP	60	22 → 55447 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
109	0.002539575	10.0.2.15	10.0.2.6	TCP	54	55447 → 22 [RST] Seq=1 Win=0 Len=0
110	0.002615468	10.0.2.6	10.0.2.15	TCP	60	22 → 55448 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
111	0.002622503	10.0.2.15	10.0.2.6	TCP	54	55448 → 22 [RST] Seq=1 Win=0 Len=0
112	0.002652818	10.0.2.15	10.0.2.6	TCP	54	55449 → 22 [SYN] Seq=0 Win=512 Len=0
113	0.002666662	10.0.2.15	10.0.2.6	TCP	54	55450 → 22 [SYN] Seq=0 Win=512 Len=0
114	0.002732071	10.0.2.15	10.0.2.6	TCP	54	55451 → 22 [SYN] Seq=0 Win=512 Len=0
115	0.002747450	10.0.2.15	10.0.2.6	TCP	54	55452 → 22 [SYN] Seq=0 Win=512 Len=0
116	0.002759132	10.0.2.15	10.0.2.6	TCP	54	55453 → 22 [SYN] Seq=0 Win=512 Len=0

Obr. 5.3: Wireshark zachytávajúci komunikáciu

Na obr. 5.3 je vidieť veľké množstvo posielaných SYN paketov z IP adresy útočníka na adresu obeti, kedy vždy po zaslaní odpovedi paketom SYN/ACK je komunikácia zo strany útočníka zahodená a útočník znovu zaplavuje obeť paketmi s novými žiadosťami o zahájenie komunikácie. Obeť napokon nebola schopná pripojiť sa na žiadnu službu a bola ochromená.

5.3 Spectre

Je typ zraniteľnosti založenej na princípe *speculative execution* – procesor kvôli zvýšeniu svojej výkonnosti háda, akou cestou sa budú vykonávať príkazy a predčasne ich vykonáva. Napríklad: tok riadenia programu závisí na hodnote necachovanej a uloženej v externej fyzickej pamäti. Keďže je táto pamäť omnoho pomalejšia, než procesor, zaberie veľa času, než sa táto hodnota stane známou – procesor teda namiesto čakania začne hádať smer riadenia, zapamätá si bod stavu registra a začne speculative execution programu na odhadovanej ceste. Ak hodnota konečne dorazí z pamäti, procesor skontroluje jeho odhad. Ak bol nesprávny, zničí vykonávaný proces vyvrátením stavu registra do pôvodného stavu. Avšak, výsledky týchto výkonov zostávajú v mikroarchitektálnej časti (cache), čím vzniká zraniteľnosť.

Spectre teda dokáže pomocou vykonania speculative executions oklamať procesor na vykonanie programov s nesprávnymi inštrukciami. Tieto pokyny nazývame prechodné. Ovplyvňovaním týchto prechodných pokynov útočník dokáže vyvolať únik informácií z pamäti obeti a tým vie preniknúť do arbitrárnej pamäti jadra [20].

Pomocou programu voľnoprístupnému na internete sa vyskúšala zraniteľnosť type Spectre aj na vývojovú dosku. Vykonanie útoku nebolo úspešné – nedokázal prechádzať cache pamäť procesoru ako je znázornené na výpise 5.5.

Výpis 5.5: Spectre

```
Reading at malicious_x = 0xffffffab0... Unclear: 0xFF=? score 999
Reading at malicious_x = 0xffffffab1... Unclear: 0xFF=? score 999
Reading at malicious_x = 0xffffffab2... Unclear: 0xFF=? score 999
Reading at malicious_x = 0xffffffab3... Unclear: 0xFF=? score 999
Reading at malicious_x = 0xffffffab4... Unclear: 0xFF=? score 999
Reading at malicious_x = 0xffffffab5... Unclear: 0xFF=? score 999
```

5.4 Brute Force hesla pre SSH

Pre jednoduchšie realizovanie útokov sa zvolila iná konfigurácia prostredia – namiesto Linux distribúcie Ubuntu 20.01.1 sa nasadila na miesto útočníka Kali Linux, ktorý disponuje širokým arzenálom nástrojov napomáhajúcim k útokom. Sieťové nastavenia virtuálnych systémov Debian a Kali v prostredí Oracle VirtualBox sa upravili na `Bridged Adapter`. Oba systémy sa nachádzajú v rovnakej sieti 192.168.0.0/24 – Debian s IP adresou 192.168.0.100 a Kali s IP adresou 192.168.0.101.

Pri skenovaní nástrojom Nmap zo systému Kali sa zistilo, že sa na rovnakej sieti nachádza zariadenie (Debian) s otvoreným portom 22, na ktorom beží služba SSH a taktiež port 443 so službou https. Pri skúmaní portu sa zistila aktívna webová aplikácia na adrese `https://192.168.0.100`. Úvodná stránka ponúkla menu

pre prihlásenie už zaregistrovaného užívateľa a možnosť vzdialeného prihlásenia pomocou linku odkazujúceho na externú stránku. Pri bližšom skúmaní zdrojového kódu pomocou Inspect módu webového prehliadača sa pri inicializácii hlavičky stránky zistila hodnota pre `request.setRequestHeader` CzechLight Webmin, ako je ukázané na výpise 5.6. Po ďalšom skúmaní prostredníctvom webového prehliadača sa zistilo, čo CzechLight predstavuje a či tieto informácie dokážu pomôcť útočníkovi.

Výpis 5.6: žiadosť hlavičky

```
request.open('GET', url);
request.setRequestHeader('X-Requested-With', 'CechLight Webmin');
request.send();
```

Keďže dostupné zariadenie má povolenú službu SSH, skúsila sa využiť metóda Brute Force na heslo tohto zariadenia. Je malá pravdepodobnosť toho, že by útočníkovi reálne napadlo využiť string CzechLight, ktorý sa ukrýval v požiadavke hlavičky ale tá možnosť tu je a chcela som ňou ukázať, aké by bolo ľahké heslo uhádnuť a získať plný prístup ku serveru, na ktorom webová aplikácia beží.

Ako nástroj pre vytvorenie slovníka sa využil `crunch` – presnejšie príkazom 5.7 obsahujúcim minimálne a maximálne množstvo znakov, možnosťou `-t`, ktorá špecifikuje základný string, z ktorého sa heslo bude vyvíjať spolu so znakmi `@` určujúcimi ich umiestnenie. Výsledok tohto príkazu sa na koniec uloží do predom vytvoreného textového súboru.

Výpis 5.7: Crunch príkaz

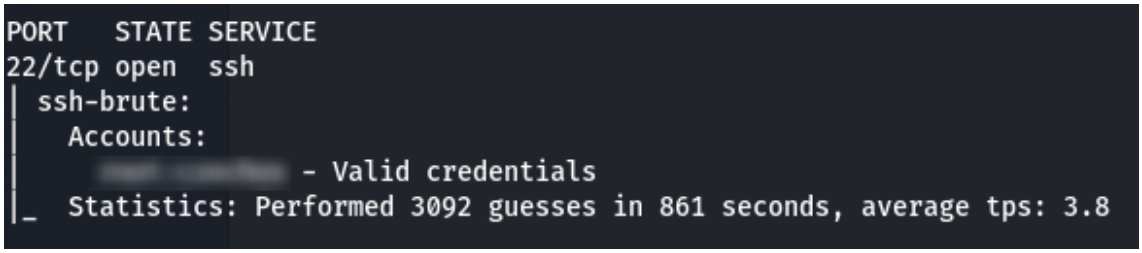
```
$ cruch 7 7 -t czech@@ > potPass.txt
Crunch will now generate the following amount of data: 5408 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 676
```

Takto vytvorený slovník potencionálnych hesiel bol využitý pri samotnom brute force útoku. Ten sa realizoval prostredníctvom Nmap skriptu na výpise 5.8 pozostávajúceho z IP adresy obeti, portu, druhu skriptu s názvom `ssh-brute` a argumentov `userdb` a `passdb` vytvorených dopredu útočníkom. V `userdb` sa nachádzajú mená užívateľov, ktorí by sa mohli nachádzať na vzdialenom serveri a v `passdb` sa nachádza textový súbor s heslami vytvorenými prostredníctvom crunch príkazu.

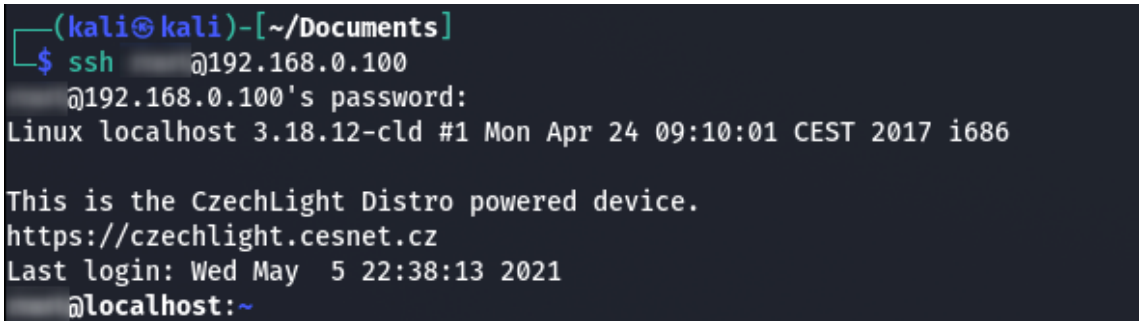
Výpis 5.8: Nmap Brute Force attack

```
$ nmap 192.168.0.100 -p 22 --script ssh-brute --script-args
\ userdb=user.txt,passdb=potPass.txt
```

Po úspešnom dokončení príkazu na obr. 5.4 skript vypísal užívateľské meno a heslo, ktoré sa mu podarilo spárovať. S týmito údajmi sa aj v skutočnosti podarilo pripojiť pomocou SSH na vzdialený server na obr. 5.5 a tým získať plný prístup aj ku aplikácii, ktorá beží na porte 443.



Obr. 5.4: Úspešné spárovanie užívateľského mena a hesla

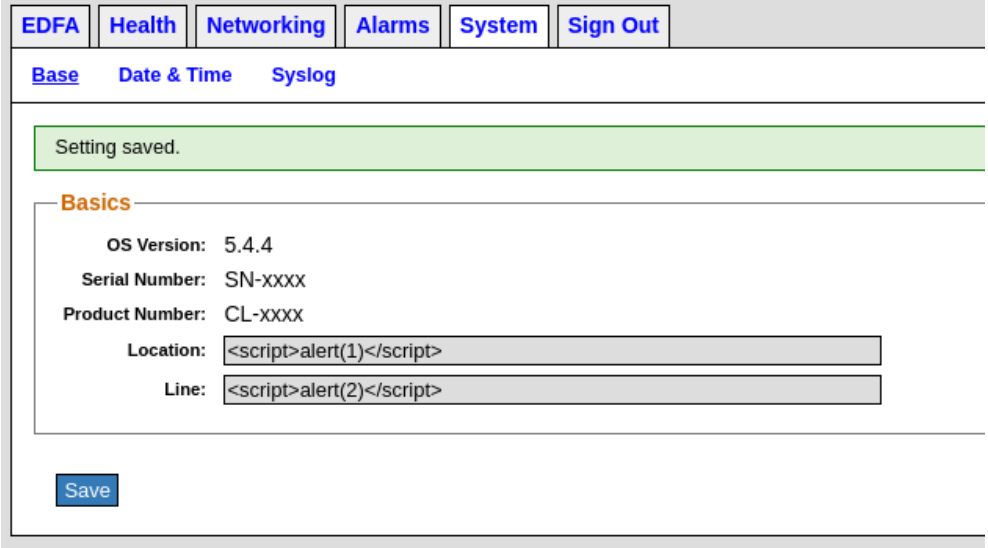


Obr. 5.5: SSH session na vzdialený server

5.5 Cross-site scripting (XSS)

Web aplikácia bežiaca na IP adrese 192.168.0.100 má po prihlásení užívateľa možnosť vloženia textu do rôznych polí. Pri tejto skutočnosti sa ponúkala využiť ponuka vloženia XSS skriptu. XSS – cross-site scripting – je typ útoku, ktorý dokáže vložiť škodlivý kód na stránku a tá ho berie ako validný vstup. Pri úspešnom vykonaní skript dokáže kompromitovať užívateľské účty, modifikovať obsah stránky či ukradnúť session cookie reálneho užívateľa a vydávať sa za neho.

Na obr. 5.6 je vidieť možnosť vloženia XSS skriptu do polí `Location:` a `Line:`. Tento skript je veľmi jednoduchý – poslúžil len pre zistenie, či je aplikácia odolná. Po uložení sa skript premietol na záložku v prehliadači ako je znázornené na obr. 5.7. Po preskúmaní zdrojového kódu na obr. 5.8 je vidieť, že vstup zadaný do polí je ošetrený prevedením znakov `< >` na `<` a `>` a je teda odolný voči XSS útoku.



Obr. 5.6: Polia po vložení XSS skriptu

Obr. 5.7: Záložka s XSS skriptom

```
<title>

        &lt;script&gt;alert(1)&lt;/script&gt;
        (&lt;script&gt;alert(2)&lt;/script&gt;) |
    Czech Light
</title>
```

Obr. 5.8: Zdrojový kód s ošetreným vstupom z polí

6 Ansible

Ansible predstavuje softvérový nástroj určený pre automatizáciu správy viacerých zariadení. Používa sa pre nasadenie aplikácií do systému, manažuje konfigurácie zariadení, aktualizuje nástroje – prostredníctvom neho sa dá jednoducho dohliadať a spravovať systémy na diaľku. Veľká výhoda spočíva v jeho jednoduchosti – ku svojej funkčnosti nepotrebuje prídavných agentov a ani extra bezpečnostné prvky – je teda ľahko spustiteľný aj na primitívnejších systémoch.

6.1 Princíp funkčnosti

Pracuje na princípe rozdelenia systémov do dvoch skupín, a to: kontrolný uzol a kontrolované uzly. Kontrolný uzol predstavuje zariadenie (PC), ktoré obsahuje ansible a musí byť aspoň jeden. Kontrolované uzly sú akékoľvek zariadenia, ktoré podliehajú manažovaniu kontrolného uzla. Kontrolný uzol sa pripája na kontrolované uzly (klient, server) na sieti a posiela na ne vytvorené konfigurácie. Posielajú sa prostredníctvom SSH spojenia – najlepšie s dopredu nastavenými SSH kľúčmi.

Moduly konfigurácií využívané pre automatizáciu pozostávajú z inštrukcií pre špecificky určené skupiny kontrolovaných uzlov alebo pre jednotlivcov. Tieto moduly sa skladajú do tzv. playbookov – ktoré sa vykonajú na kontrolovanom uzle. Jazyk, ktorým sa konfigurujú sa nazýva `YAML`. Má intuitívnu vizuálnu štruktúru a jeho logika je jednoduchá na pochopenie. Pozostáva z dvoch druhov dátových blokov: sekvencie a mapovanie. Sekvencie predstavujú hodnoty uvedené v určitom poradí. Začínajú sa znakom `-` nasledovaný medzerou. Dajú sa prirovnať ku listu v jazyku Python alebo k polu v Bash. Mapovanie značí dvojicu kľúča a jeho hodnotu. Každý kľúč musí byť unikátny a nezáleží na jeho poradí [21].

6.2 Príprava systémov

Táto sekcia dokumentu sa zaoberá prípravou na inštaláciu nástroja ansible a prispôsobovania systému. Pretože systém Debian už nieje podporovaný, opomenú sa aktualizácie systému a nefunkčnosť rôznych nástrojov.

6.2.1 Debian server – kontrolovaný uzol

Pre fungovanie nástrojov bolo nutné pozmeniť zoznam zdrojových webových stránok v `/etc/apt/source.list` ako je ukázané na výpise 6.1.

Výpis 6.1: source.list

```
deb http://http.debian.net/debian jessie main contrib
deb-src http://http.debian.net/debian jessie main contrib

deb http://security.debian.org jessie/updates main contrib
deb-src http://security.debian.org jessie/updates main contrib
```

Po prispôsobení zdrojov sa ďalej musela nainštalovať novšia verzia Python interpretéra, presnejšie verzia 3.7.3. Začať sa muselo inštalovaním esenciálnych balíčkov potrebných k vytvoreniu Python zdroja, a to príkazom `sudo apt install build-essential zlib1g-dev libncurses5-dev libgdbm-dev libnss3-dev libssl-dev libreadline-dev libffi-dev curl libbz2-dev`. Po úspešnom inštalovaní nasledovalo samotné stiahnutie Python s príkazom `curl -O https://www.python.org/ftp/python/3.7.3/Python-3.7.3.tar.xz`. Pokračujeme s extrahovaním stiahnutého súboru s `tar -xf Python-3.7.3.tar.xz`. Ďalej sa presunieme do adresára práve extrahovaného Pythonu `cd Python-3.7.3` a spustíme skript, ktorý skontroluje, či systém obsahuje všetky potrebné závislosti `./configure -enable-optimizations`. Ak kontrola prebehla úspešne, môže nasledovať zostavovací proces `make -j 1`, kde 1 značí počet jadier procesoru. Akonáhle je zostavovanie dokončené s inštaláciou Python binárnych súborov, nasleduje samotná inštalácia s `make altinstall` – namiesto klasického `install` je použitá druhá varianta, aby sa neprepísali pôvodné binárne súbory Python.

6.2.2 SHH autentifikácia

Ako ďalší krok nasleduje konfigurovanie SSH autentifikácie. Pre účely bakalárskej práce sa zvolila možnosť prihlasovania pomocou užívateľa `root`, aj keď je to považované ako zraniteľné – neodporúča sa takto postupovať budúcemu užívateľovi týchto systémov (v nadchádzajúcich častiach sa ale vytvorí nový `sudo` užívateľ pre bezpečnejšiu prácu). Keďže `root` užívateľ už vytvorený je, môže sa pokračovať na vygenerovanie verejného a privátneho kľúča na systéme Kali. Toto zabezpečuje príkaz `ssh-keygen -b 4096`, ktorý vytvorí rsa kľúče o veľkosti 4096 bitov (je možné použiť aj kratšie kľúče s algoritmom ed25519) [22]. Po stlačení enteru sa objaví možnosť ponechania pôvodných názvov súborov, v ktorých sú uložené `id_rsa` a `id_rsa.pub` v adresári `/home/root/.ssh/id_rsa` . Pri vytváraní kľúčov je možnosť nastavenia dodatočného hesla `passphrase` pre pridanú hodnotu autentifikácie, bezktorého potom nieje možné samotné zahájenie SSH pripojenia na vzdialené zariadenie. Ako ďalší krok sa vytvorený verejný kľúč skopíruje na vzdialený systém – Debian. Najprv sa započne spojenie cez`ssh root@192.168.0.100`. Ďalej sa vytvorí nový adresár a ak neexistuje, tak aj súbor `authorized_keys` s pomocou `mkdir`

`-p ~/.ssh && touch ~/.ssh/authorized_keys` príkazu. Novo vytvorenému adresáru a súboru priradíme nové práva `chmod 700 ~/.ssh && chmod 600 ~/.ssh/authorized_keys`. Ako posledný krok sa skopíruje vytvorený verejný kľúč užívateľa z Kali na vzdialené zariadenie prostredníctvom príkazu `scp ~/.ssh/id_rsa.pub root@192.168.0.100:~/.ssh/authorized_keys`. Nakoniec sa užívateľ skúsi pripojiť z Kali cez `ssh root@192.168.0.100` a po zadaní hesla by mal mať prístup na vzdialený server.

6.2.3 Kali – kontrolný uzol

Keďže Kali a Debian už mali nastavené fungujúce pripojenie z predchádzajúcich útokov, pokračovala som vo využívaní tohto systému ako kontrolný uzol, z ktorého sa bude riadiť vzdialený Debian. Nástroj ansible sa teda inštaloval na Kali zariadenie s klasickým príkazom `sudo apt-get install ansible`. Po inštalácii sa objaví v `/etc/ansible` jeho konfiguračný súbor `ansible.cfg`, súbor `hosts` – doň sa vkladajú IP adresy jednotlivcov alebo skupín označených v hranatých zátvorkách – ukážka je znázornená na obr. 6.1.

```
[linuxMachines]
192.168.0.100
```

Obr. 6.1: Ukážka súboru hosts s IP adresou vzdialeného zariadenia

V konfiguračnom súbore sa odkomentovala cesta k už spomínanému súboru `hosts`, ďalej sa umožnilo využívanie `root` užívateľa, port, cez ktorý prebieha SSH a teda aj prenos súborov ansible. Môžeme odkomentovať aj možnosť `host_key_checking = False`, ktorá kontroluje kľúče užívateľov pri SSH. `remote_user = root` sa taktiež odkomentoval pre uľahčenie práce, keďže root užívateľ má najvyššie práva a vykonáva všetko bez nutnosti pridania práv. Taktiež, ak by si užívateľ neprial, môže znovu zakomentovať `log_path = /var/log/ansible.log`. Pre informovanosť sa umožnila možnosť zobrazenia hostov `display_skipped_hosts = True`, ktorí sú preskočení, pretože sa ich daná úloha netýka. Znovu pre uľahčenie sa v sekcii `[privilege_escalation]` umožnili možnosti `become=True` a `become_user= root`.

Pre ansible je vhodné mať nástroje uložené ako role, teda v adresári `roles`. Hlavný skript, ktorý sa bude spúšťať a bude vykonávať všetky inštrukcie sa nazýva `ansibleScript.yml` a spúšťa sa príkazom `ansible-playbook ansibleScript.yml`.

Opísaná štruktúra vyzerá následovne:

```
/etc/
└── ansible/
    ├── ansible.cfg
    ├── ansibleScript.yml
    ├── hosts
    └── roles/
        ├── fail2ban
        └── newSudoUser
```

Samotný skript `ansibleScript.yml` zahŕňa výpis 6.2:

Výpis 6.2: ansibleScript.yml

```
- name:  improved configuration of Linux Debian
  hosts: linuxMachines

  roles:
    - fail2ban2
    - newSudoUser
```

Obsah skriptu pozostáva z názvu pre lepšiu orientáciu, hostov, na ktoré sa má tento skript aplikovať a rolí, ktoré obsahujú konkrétne konfigurácie fail2ban a newSudoUser.

6.2.4 Fail2ban

Fail2ban predstavuje účinný nástroj zamedzujúci útokom na autentifikáciu napr. SSH služby. Základný princíp je monitorovanie `log` súborov a sledovanie výskytu určitého modelu v autentifikačných zlyhaní. Po konfigurovaní filtru na určitú službu fail2ban identifikuje model prostredníctvom `failregex`, ktorý v jednoduchosti obsahuje výraz, ktorý filter v `log` súbore hľadá [23].

Ak filter nájde zhodu, prichádza na rad predom definovaná akcia, ktorá sa vykoná. Administrátor si `action` vie prispôsobiť potrebám systému alebo svojim preferenciám. Vo východzom stave ale `action` vyhostí IP adresy modifikovaním tabuľky `iptables` pravidiel firewallu. V konfigurácii sa dá nastaviť aj posielanie emailov s podrobnejšími informáciami.

Predvolené nastavenia pre akciu po úspešnom detekovaní zamietnutých prihlásení pozostávajú z 3 zamietnutých autentifikácií uskutočnených v priebehu 10 min a vyhostenia na dobu 10 min. Tieto nastavenia ale nahradia nastavenie pre SSH premávku, kde vyhostenie prebieha až po šiestich zamietnutých autentifikáciach.

Pri pôvodných nastaveniach `iptables` pre SSH premávku fail2ban vytvorý nový reťazec pri zahájení spojenia a pridá nové pravidlo do `INPUT` reťazca, kam sa odosiela všetka TCP premávka pre port 22, aplikuje sa pravidlo a premávka sa znovu vráti do pôvodného reťazca. Toto nastane v prípade, ak pravidlo nič nenašlo. V opačnom prípade sa pravidlo pridá na vrch nového reťazca a premávka sa zruší – nastáva vyhostenie. Príklad tabuľky s pridanými pravidlami je na obr. 6.2.

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain fail2ban-nginx-http-auth (0 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere

Chain fail2ban-ssh (0 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere

Chain fail2ban-ssh-ddos (0 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
```

Obr. 6.2: IP tables

Štruktúra role fail2ban pozostáva z:

```
/roles
└── /fail2ban2/
    ├── defaults/
    │   └── main.yml
    ├── handlers/
    │   └── main.yml
    ├── meta/
    │   └── main.yml
    ├── README.md
    ├── tasks/
    │   └── main.yml
    ├── templates/
    │   ├── jail.local.j2
    │   └── fail2ban.local.j2
    ├── tests/
    │   ├── inventory
    │   └── test.yml
    └── vars/
        └── main.yml
```

Role sa dajú vygenerovať pomocou príkazu `ansible-galaxy init <meno role>`. Takto vznikne adresár s podadresármi defaults, handlers, meta, tasks, templates, tests a vars. Defaults obsahuje súbor `main.yml` v ktorom sa nachádzajú hodnoty premenných z adresára templates. V handlers sa nachádzajú väčšinou procedúry reštartu služieb. Meta obsahuje informácie o modulu galaxy. README.md je textový súbor s krátkym opisom role a iných prídavných informácií potrebných ku správnemu fungovaniu rolí. Tasks zahŕňa skript, ktorý obsahuje menšie metódy, ktoré sa starajú o správne inštalovanie fail2ban na kontrolovaný uzol, kopírovanie všetkých potrebných súborov a ich implementovanie na kontrolný uzol. V templates sú súbory typu jinja2, ktoré priamo obsahujú vylepšenú konfiguráciu a kopírujú sa prostredníctvom taskov na kontrolný uzol. Kopírovanie týchto súborov je veľmi dôležité, pretože ak by sa nechali konfigurácie v pôvodných súboroch, ktoré dodal fail2ban, tak by sa pri každom aktualizovaní zmazali a už by neexistovali. Sú tu teda konfigurácie priamo pre fail2ban nástroj a pre súbor s pravidlami pre vyhostenie IP adries. V tejto roli sa nachádza aj adresár tests, ktorý sa ale nevyužil, rovnako tak ako aj vars.

6.2.5 newSudoUser

Rola `newSudoUser` má rovnakú štruktúru ako fail2ban ale narozdiel od nej je menšia. Obsahuje defaults s hodnotami premenných z adresára templates. Jedná sa o hodnoty, ktoré konfigurujú SSH službu v konfiguračnom súbore `/etc/ssh/sshd_config`. Rola každopádne funguje rovnako ako predchádzajúca, teda súbory z templates sa cez tasks prekopírujú na kontrolný uzol spolu s hodnotami z defaults a vykonajú nastavené konfigurácie.

Ďalšou úlohou tejto role je vytvoriť nového `sudo` užívateľa, pretože na kontrolnom uzle sa nachádza len jeden užívateľ `root`. Je odporúčané ovládať systém užívateľom so `sudo` právami a nie priamo `root`, aby sa predišlo unáhleným či neopatrným krokom, ktoré môžu mať fatálne dopady. Taktiež sa nastavovalo prihlasovanie na tohto užívateľa z kontrolného uzlu pomocou SHH autentifikácie verejného a privátneho kľúča. Na obr. 6.3 je znázornené prihlásenie na vytvoreného užívateľa `test2` z kontrolného uzla. Ako je vidieť, prihlásenie prebehlo bez potreby zadávať užívateľské heslo, čo znamená, že verejný kľúč užívateľa bol úspešne pridaný na kontrolný uzol.

```
┌──(kali㉿kali)-[~]
└─$ ssh test2@192.168.0.100
Linux localhost 3.18.12-cld #1 Mon Apr 24 09:10:01 CEST 2017 i686

This is the CzechLight Distro powered device.
https://czechlight.cesnet.cz
Last login: Mon May 10 16:24:00 2021 from 192.168.0.101
test2@localhost:~$
```

Obr. 6.3: Sudo užívateľ test2

Štruktúra role newSudoUser:

```
/roles
└── /newSudoUser/
    ├── defaults/
    │   └── main.yml
    ├── handlers/
    │   └── main.yml
    ├── meta/
    │   └── main.yml
    ├── README.md
    ├── tasks/
    │   └── main.yml
    ├── templates/
    │   └── ssh-config-improved.j2
    ├── tests/
    │   ├── inventory
    │   └── test.yml
    └── vars/
        └── main.yml
```

7 Stagnácia systému

Systému Linux Debian Jessie vypršala dĺžka podpory v lete roku 2020, preto pri skúšaní aplikovaní ďalších bezpečnostných vylepšení bolo veľmi veľa problémov a bolo by sofistikovanejšie najprv tento systém aktualizovať na Debian Stretch a na túto platnú distribúciu skúsiť uplatniť vytvorené skripty. Na Debian Jessie sa totiž nevydávajú nové aktualizácie, takže nebolo možné uplatniť rôzne odporúčania, ktoré sú uvedené v teoretickej časti práce a taktiež nebolo možné aplikovať efektívnu ochranu voči úspešným útokom.

7.1 Kryptovanie disku

Jeden z úspešných útokov bol aj password recovery, ktorý spočíva v zmene hesla `root` užívateľa pri bootovaní systému. Takémuto typu útoku sa dá predchádzať zaheslovaním určitej časti disku, ktorá sa načíta pri bootovaní. Zaheslovaním tejto partície sa spustí len tá časť jadra, ktorá nám umožní zadať nami nastavené heslo a iba týmto spôsobom sa dá dostať ďalej do systému.

Pri skúmaní toho, ako by sa disk mohol zaheslovať sa narazilo na veľa problémov svtým, že kryptovanie sa dalo nastaviť len fyzicky pri inštalácii distribúcie alebo pomocou open-source nástrojov, ktoré ale neboli podporované distribúciou Jessie. Ďalšia možnosť bolo zálohovanie všetkých dát z Jessie a skúsiť aktualizovať systém na Stretch, čo by ale vlastne už nebol zastaralý systém, ako je uvedené vvzadaní bakalárskej práce.

7.2 Zastaralé certifikáty

Na zariadení sa nachádza aplikácia, ktorá nemá aktuálne bezpečnostné certifikáty. Zastaralé certifikáty by sa dali nahradiť overenými certifikátmi prostredníctvom certifikačných autorít ako napr. `Let's Encrypt` a jeho klienta `certbot`. Toto ale nefunguje znovu kvôli zastaralému Debian Jessie.

Záver

V bakalárskej práci sa zaoberalo Linuxovým systémom s distribúciou Debian Jessie 8.9 a verziou jadra 3. 18. 12-cld na vývojovej doske. Opisovali sa teoretické poznatky o systéme Linux, druhy útokov aplikovateľných na systém, preberali sa potenciálne zraniteľnosti a povrchne navrhované vylepšenia a napokon sa využili dva druhy sken programov ako Lynix a Nmap ako nápomoc pri odhaľovaní zraniteľností. Výstupom je dôkaz – útok, že je systém zraniteľný a je potrebné ho podrobnejšie nakonfigurovať a zabezpečiť voči iným potenciálnym útokom.

Na daný systém sa podarilo úspešne aplikovať útoky ako password recovery, ktorým sa docielilo prostredníctvom priameho fyzického prístupu ku systému zmeniť heslo najdôležitejšieho užívateľa – root a tým získať plnohodnotný prístup, či Denial of Service, presnejšie typu SYN flood, kedy sa za použití externých softvérových nástrojov dokázalo vykonať zaplavenie adresy obeti SYN paketmi a tým pádom odopretiu služieb. Ďalší úspešný útok je brute force aplikovaný na heslo užívateľa zariadenia. Bol pokus útočiť aj priamo na jadro útokom Spectre, ktorý sa ale nepodaril aplikovať úspešne. Ak by ale úspešný bol, dokázal by si zabezpečiť prístup do pamäti jadra. Pri skúmaní aplikácie sa skúšal útok cross-site scripting, ktorý sa ale taktiež nestretol s úspechom.

Boli nasadené ochranné prvky proti týmto útokom v podobe skriptu, ktorý je distribuovaný prostredníctvom nástroja Ansible, ktorý automatizuje správu zariadení. Do skriptov sú nasadené role, v ktorých spočíva ochrana zraniteľného systému. Rola Fail2ban sa zaoberá zaradením nových pravidiel do firewallu, ktoré na základe vymedzených pravidiel blokujú IP adresy útočníkov. Druhá rola sa nazýva newSudoUser, ktorá už zo svojho názvu prezrádza jej cieľ – vytvoriť nového užívateľa so sudo právami spolu s ssh autentifikáciou prostredníctvom verejného kľúča. V tejto roli sa taktiež nachádza konfigurácia, ktorá upravuje SSH nastavenia.

Kvôli limitácii zo strany zastaralého operačného systému sa vyskytli komplikácie a nebolo možné vytvoriť efektívne riešenia pre zabránenie iným útokom. Preto sa v práci nachádzajú odporúčania, ktoré opisujú, ako by sa malo ďalej so systémom pokračovať.

Literatúra

[1] SILBERSCHATZ, Abraham, Peter Baer GALVIN a Greg GAGNE. *Operating System Concepts*. 10. vydanie. New Jersey, USA: Wiley, 2018. 1278 s. ISBN 978-1-119-32091-3.

[2] KARBAN, Karel. *UNIX pro začátečníky.* 1. vydanie. Praha: Plus, 1992. 117 s. ISBN 80-85297-35-3.

[3] NEŠVERA, Šimon, Karel RICHTA a Petr ZEMÁNEK. *Úvod do operačního systému UNIX*. 1. vydanie. Praha: České vysoké učení technické v Praze, 1991. 182 s. ISBN 80-01-00592-5.

[4] GARCARZ, Mike. *The unix philosophy*. 1. vydanie. Newton, MA, USA: Digital press, 1995. 87 s. ISBN 1-55558-123-4.

[5] STALLMAN, Richard. *Linux and the GNU System* [online]. 1997, posledná aktualizácia 30. 12. 2019 [cit. 18. 11. 2020]. Dostupné z URL: <`https://www.gnu.org/gnu/linux-and-gnu.en.html`>.

[6] CASTRO, Jose Dieguez. *Introducing Linux Distros: Choose the right Linux distribution for your needs*. 1. vydanie. Spain: Apress, 2016. 366 s. ISBN 978-1-4842-1393-3.

[7] Penta Security *What Are Session Replay Attacks?* [online]. 2020, posledná aktualizácia 17. 7. 2020 [cit. 2. 12. 2020]. Dostupné z URL: <`https://www.pentasecurity.com/blog/session-replay-attacks/`>.

[8] CUPPENS-BOULAHIA, Nora, Frédéric CUPPENS a Joaquin GARCIA-ALFARO. *Data and Applications Security and Privacy XXVI*. 1. vydanie. France, Paris: Springer, 2012. 340 s. ISBN 978-3-642-31539-8.

[9] STARLINGS, William. *Network security essencials: Applications and standarts*. 4. vydanie. New Jersey, USA: Pearson Education, 2010. 432 s. ISBN 13: 978-0-13-610805-4.

[10] CISA. *Understanding Denial-of-Service Attacks* [online]. 2009, posledná aktualizácia 20. 11. 2019 [cit. 9. 12. 2020]. Dostupné z URL: <`https://us-cert.cisa.gov/ncas/tips/ST04-015`>.

[11] SWINHOE, Dan. *What is a man-in-the-middle attack? How MitM attacks work and how to prevent them* [online]. 2019, posledná aktualizácia 13. 2. 2019 [cit. 9. 12. 2020]. Dostupné z URL: <https://www.csoonline.com/article/3340117/what-is-a-man-in-the-middle-attack-how-mitm-attacks-work-and-how-to-prevent-html>.

[12] Adrian. *Active and Passive Attacks: Differences and prevention* [online]. 2020, posledná aktualizácia 10. 7. 2020 [cit. 9. 12. 2020]. Dostupné z URL: <https://www.internetsecurity.tips/active-and-passive-attacks-differences-and-prevention/>.

[13] KALSI, Tajinder. *Practical Linux Security Cookbook*. 2. vydanie. Birmingham, UK: Packt Publishing, 2018. 482 s. ISBN 978-1-78913-839-9.

[14] HASAN, Musaab a Zayed BALBAHAITH. *Mastering Linux Security: Step by Step Practical Guide*. 1. vydanie. Mauritius: LAP LAMBERT, 2020. 124 s. ISBN 978-620-0-56603-4.

[15] TEVAULT, Donald A. *Masterring Linux Security and Hardening*. 2. vydanie. Birmingham, UK: Packt Publishing, 2020. 652 s. ISBN 978-1-83898-177-8.

[16] VANNEY, Ivan. *Disable Unnecessary Sevices Debian Linux*. Linux Hint [online]. 2020 [cit. 06. 12. 2020]. Dostupné z URL: <https://linuxhint.com/disable_unnecessary_services_debian_linux/>.

[17] Marcel. *12 Critical Linux Log Files You Must be Monitoring*. Eurovps [online]. 2018, posledná aktualizácia 19. 04. 2018 [cit. 06. 12. 2020]. Dostupné z URL: <https://www.eurovps.com/blog/important-linux-log-files-you-must-be-monitoring/>.

[18] BRADFORD, Contel. *Linux Backup Types and Tools Explored*. StorageCraft: Recovery Zone [online]. 2020, posledná aktualizácia 13. 09. 2020 [cit. 06. 12. 2020]. Dostupné z URL: <https://blog.storagecraft.com/linux-backup-types-tools-explored/>.

[19] *Cisofy: Lynis documentation*. [online]. [cit. 06. 12. 2020]. Dostupné z URL: <https://cisofy.com/documentation/lynis/get-started/#first-run>.

[20] KOCHER, P., Horn J., Fogh A., Genkin D., Gruss D., Haas W., Hamburg M., Lipp M„ Mangard S., Prescher T., Schwarz M., Yarom Y *Spectre Attacks: Exploiting Speculative Execution*. In: Spectreattack [online]. 2018 [cit. 11. 12. 2020]. Dostupné z URL: <https://spectreattack.com/spectre.pdf>.

[21] HEIDI, Erika. *An Introduction to Configuration Management with Ansible* [online]. 2020, posledná aktualizácia 15. 05. 2020 [cit. 20. 05. 2021]. Dostupné z URL: <https://www.digitalocean.com/community/conceptual_articles/an-introduction-to-configuration-management-with-ansible>.

[22] BRENDEL J., Cremers C., Jackson D., Zhao M. *The Provable Security of Ed25519: Theory and Practice* [online]. 2021 [cit. 20. 05. 2021]. Dostupné z URL: <https://publications.cispa.saarland/3238/>.

[23] ELLINGWOOD, Justin. *How To Protect SSH with Fail2Ban on Ubuntu 14.04* [online]. 2020, posledná aktualizácia 07. 05. 2014 [cit. 20. 05. 2021]. Dostupné z URL: <https://www.digitalocean.com/community/tutorials/how-to-protect-ssh-with-fail2ban-on-ubuntu-14-04>.

A Obsah prílohy

Priložený skript je potrebné len skopírovať do správneho adresára v ansible a prípadne si ho upraviť podľa potrieb.

```
/ansible/
├── ansible.cfg
├── ansibleScript.yml
├── hosts
└── roles/
    ├── fail2ban2/
    │   ├── defaults/
    │   │   └── main.yml
    │   ├── handlers/
    │   │   └── main.yml
    │   ├── meta/
    │   │   └── main.yml
    │   ├── README.md
    │   ├── tasks/
    │   ├── main.yml
    │   ├── templates/
    │   │   ├── fail2ban.local.j2
    │   │   └── jail.local.j2
    │   ├── tests/
    │   │   ├── inventory
    │   │   └── test.yml
    │   └── vars/
    │       └── main.yml
    └── newSudoUser/
        ├── defaults/
        │   └── main.yml
        ├── handlers/
        │   └── main.yml
        ├── meta/
        │   └── main.yml
        ├── README.md
        ├── tasks/
        ├── main.yml
        ├── templates/
        │   └── ssh-config-improved.j2
        ├── tests/
        │   ├── inventory
        │   └── test.yml
        └── vars/
            └── main.yml
```